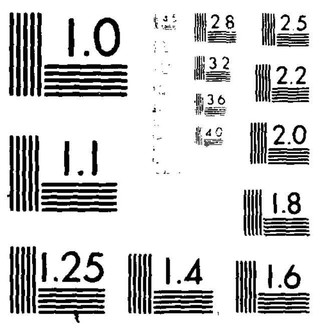
AD-A097 029 GENERAL ELECTRIC CO ARLINGTON VA INFORMATION SYSTEMS--ETC F/G 9/2
THE EFFECTS OF THE SYMBOLOGY AND SPATIAL ARRANGEMENT OF SOFTWAR--ETC(U)
FEB 81 S B SHEPPARD, E KRUESI N00014-79-C-0595
UNCLASSIFIED TR-81-388200-3 NL

1 OF 1
AD
A097029

END
DATE
FILMED
4-81
DTIC



MICROCOPY RESOLUTION TEST CHART

LEVEL III

(12)

THE EFFECTS OF THE SYMBOLOGY AND SPATIAL ARRANGEMENT OF SOFTWARE SPECIFICATIONS IN A CODING TASK

SYLVIA B. SHEPPARD

ELIZABETH KRUESI

SOFTWARE MANAGEMENT RESEARCH
INFORMATION SYSTEMS PROGRAMS
GENERAL ELECTRIC COMPANY
1755 JEFFERSON DAVIS HIGHWAY
ARLINGTON, VIRGINIA, 22202

TR-81-388200-3
FEBRUARY 1981

GENERAL ELECTRIC

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A097029	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) The Effects of the Symbology and Spatial Arrangement of Software Specifications in a Coding Task,		5. TYPE OF REPORT & PERIOD COVERED (9) Technical Report,
		6. PERFORMING ORG. REPORT NUMBER (14) TR-81-388200-3
7. AUTHOR(s) (10) Sylvia B. Sheppard, Elizabeth Kruesi		8. CONTRACT OR GRANT NUMBER(s) (15) N00014-79-C-0595
9. PERFORMING ORGANIZATION NAME AND ADDRESS Information Systems Programs General Electric 1755 Jefferson Davis Hwy, Arlington, VA 22202		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 196-160
11. CONTROLLING OFFICE NAME AND ADDRESS Engineering Psychology Programs, Code 455 Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE (11) Feb[redacted]981
		13. NUMBER OF PAGES 39
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) same (12) 50		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

same

18. SUPPLEMENTARY NOTES

Technical Monitor: Dr. John J. O'Hare

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software engineering, Software experiments, Structured programming, Modern programming practices, Software documentation, Flowcharts, Program design language, Software human factors.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the second in a series of experiments to evaluate the effects of the format of software specifications on programmer performance. The current experiment examined performance on a coding task. Thirty-six professional programmers were presented with specifications for each of three modular-sized programs. Nine different specification formats were prepared for each program. These formats varied along two dimensions: type of symbology and spatial arrangement. The type of symbology included

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

404446

natural language, constrained language (PDL), and ideograms (flowchart symbols). The spatial arrangement included sequential (vertical flow), branching (flowchart), and hierarchical (tree-like). Working from the specifications, the participants constructed a section of code at the middle of each program. These sections contained about fifteen lines and included the most complex decision structures present in the programs. The participants were instructed to complete the code, using a text editor. When satisfied that the program would perform correctly, they submitted it for a compilation and run. If the compilation was unsuccessful or the program did not run correctly, the participants were asked to correct the errors and submit the run again.

The difficulty of the coding task was measured by four dependent variables: (1) the time to code and debug, (2) the number of submissions required for a correct run, (3) the number of errors, and (4) the number of editor transactions.

The three programs differed substantially in difficulty. An analysis of the error data revealed that these differences were due to errors in the control flow and not to errors related to assignment statements or variables. Substantial differences were also associated with the type of symbology. The natural language was considerably more difficult to code from than the constrained language or ideograms. An examination of the error data showed that these differences were due both to errors in coding the control flow and to errors related to assignment statements and variables. The effect of the spatial arrangement was not as great as the effect of symbology. Although not statistically significant, the branching arrangement appeared to be superior to the sequential and hierarchical arrangements, particularly in minimizing control-flow errors. A comparison of the individual formats revealed that the constrained language presented in a sequential or in a branching arrangement resulted in the highest level of performance.

The results of this experiment were largely consistent with those from the first experiment which examined performance on a comprehension task. It was suggested that software developers convert specifications to a PDL before coding begins.

TR-81-388200-3

THE EFFECTS OF THE SYMBOLOGY AND SPATIAL ARRANGEMENT OF SOFTWARE SPECIFICATIONS IN A CODING TASK

Sylvia B. Sheppard
Elizabeth Kruesi

Software Management Research
Information Systems Programs
General Electric Company
1755 Jefferson Davis Highway
Arlington, Virginia 22202

Submitted to:

Office of Naval Research
Engineering Psychology Programs
Arlington, Virginia

Contract: N00014-79-C-0595
Work Unit: NR 196-160

Approved by: William E. Porter
William E. Porter, Manager
Advanced Information Systems & Technology

February 1981

TABLE OF CONTENTS

INTRODUCTION

A continuing controversy surrounds the use of flowcharts as software documentation tools. Flowcharts have been described as everything from "an essential tool in problem solving" (Bohl, 1971, p.53) to "an obsolete nuisance" (Brooks, 1975, p.168). An early empirical assessment of the value of flowcharts in programming was reported by Shneiderman, Mayer, McKay and Heller (1977). They performed a series of experiments on the composition, comprehension, debugging and modification of programs. For the composition task, the participants were asked to write a program; some were also asked to produce a flowchart in addition to the program. For the comprehension, debugging, and modification tasks, all participants were given a program listing while some were given a flowchart as an additional aid. Shneiderman et al. found no significant differences in any of their experiments between groups that did and did not use flowcharts. In another study, Ramsey, Atwood, and Van Doren (1978) found no advantage for flowcharts over a Program Design Language (Caine and Gordon, 1975) for generating program designs or for translating the designs into code. In fact, the designs expressed in a Program Design Language (PDL) were judged to be of superior quality in that they included greater algorithmic detail, more modularization, and less abbreviation of variable names than those expressed as flowcharts.

Although studies performed on software-related tasks have not been favorable to flowcharts, experiments performed in other areas of information presentation have demonstrated an advantage for flowcharts over alternative presentation formats including prose descriptions, short sentences, and decision tables (Wright and Reid, 1973; Blaiwes, 1974; Kammann, 1975). Kammann, for example, presented participants with a set of telephone dialing problems. The dialing

instructions were presented in the form of a prose description or a flowchart. Fewer errors were made with the flowchart. [For a review of the non-software research, see Sheppard, Kruesi, and Curtis (1980)].

Characteristics of Software Documentation

Our approach to evaluating flowcharts and other forms of documentation is to investigate how various characteristics of presentation affect the performance of programmers on typical software-related tasks. There are two primary dimensions for categorizing how available documentation aids configure the information they present to programmers (Jones, 1979). The first dimension is the type of symbology in which information is presented. The second dimension is the spatial arrangement of this information.

Type of Symbology. The symbology dimension includes narrative text, constrained language, and ideograms. Documentation in the form of narrative text is frequently found embedded in the source code as either global or in-line comments. Constrained language, which is embodied in a Program Design Language (PDL), is more succinct than narrative text, using strictly defined keywords to describe arguments or predicates. Ideograms are frequently found in flowcharts and HIPO charts (Bohl, 1971; Katzen, 1976). A standard set of ideograms has come to represent processes or entities within a program.

Spatial Arrangement. The spatial arrangement of information in documentation is a second dimension along which documentation techniques can be categorized. In the current experiment, this dimension is represented by a sequential, a branching, and a hierarchical

arrangement. A sequential arrangement is typical of narrative descriptions, program listings and PDL while a branching arrangement is typical of flowcharts. A hierarchical arrangement is not generally used for individual module specifications but, rather, at the system level to present a visual display of the relationship among modules.

This report describes the second in a series of experiments to investigate the effects of the type of symbology and the spatial arrangement. For all experiments, the three types of symbology (narrative text, constrained language, and ideograms) are factorially combined with the three spatial arrangements to produce nine different documentation formats. The first experiment, which is described in Sheppard, Kruesi, and Curtis (1980), investigated comprehension performance. The current experiment examined the influence of these dimensions on the ability of programmers to translate the specifications into code.

Effects of Symbology and Spatial Arrangement on Comprehension

In the first experiment, seventy-two professional programmers were presented with specifications for each of three modular-sized computer programs. The participants answered a series of comprehension questions for each program using only the specifications. The questions were presented interactively on a CRT and consisted of three different types. For forward-tracing questions, the participants were given a set of conditions from the program. Their task was to trace through the specifications and find the first statement executed under those conditions. For backward-tracing questions, they were required to locate a given statement within the specifications and then determine the set of conditions which lead

to that point. For the input-output questions, they were given input data and were asked to determine the value of particular variables at a later point.

Both forward- and backward-tracing questions were answered more quickly from specifications presented in constrained language or ideograms than in natural language. Forward-tracing questions were answered most quickly from a branching arrangement and backward-tracing questions were answered more quickly from the branching and hierarchical arrangements. Response times to the input-output questions did not vary significantly as a function of the type of symbology or the spatial arrangement. It was concluded that the control structure of software was made more comprehensible by presenting the specifications in a succinct symbology (constrained language or ideograms) and in a branching arrangement. The purpose of the present experiment was to determine the optimal format for translating the specifications into code.

Coding

An important part of the software development process involves translating a set of program specifications into a working program. A number of studies have investigated the program construction process (e.g., Boies and Gould, 1974; Youngs, 1974; Lucas and Kaplan, 1974; Sime, Green, and Guest, 1977; Dunsmore and Gannon, 1978). Interest has centered on the effects of numerous factors such as the experience level of the programmers, the presence or absence of various language features, and the use or non-use of structured coding. The dependent measures have included the amount of programming time to successfully construct a program, the number of submissions required for a correct run, the relative frequency of various classes of errors, and the quality (e.g., complexity, readability) of the resulting code.

The effect of specification format on the program construction process has been examined in two studies. Sheppard, Milliman, and Curtis (1979) required professional programmers to construct three programs. An English description of each problem was presented in addition to one of the following specification formats: (1) a program design language (PDL), (2) a tree chart, or (3) both of these formats. No significant differences were found in the times required to construct programs from these different specification formats. In the study mentioned earlier, Ramsey, Atwood, and Van Doren (1978) presented specifications in the form of a flowchart or in PDL to graduate students in computer science. They found no difference between these two groups in the quality of the implemented programs. (The amount of time required was not reported.)

In the current experiment, the participants constructed programs on-line. An automated data-collection system was used which recorded the timing and the complete sequence of events involved in coding and debugging each program. The participants constructed major sections at the middle of each of three programs. These sections contained about fifteen lines of code and included the most complex decision structures present in the program. The primary questions of interest concerned differences in the time required to successfully construct the programs, in the number of submissions required, and in the number and types of errors as a function of the specification formats.

METHOD

Participants

Thirty-six professional programmers from four different locations participated in this experiment. Thirty-two were General Electric employees; the others worked for the Department of Defense. The participants averaged 5.3 years of professional programming experience and had used an average of four programming languages.

Independent Variables

The experiment was designed to study the effects of three independent variables: the type of symbology, the spatial arrangement of the information, and the type of program.

Program type. In our previous research (Sheppard, Curtis, Milliman & Love, 1979) significant differences in programmer performance were often associated with differences among programs. Three programs of varying types were chosen for use in this experiment. (The same three programs were used in the first experiment as well.) A program which calculated the trajectory of a rocket was chosen as representative of an engineering algorithm. An inventory system for a grocery distribution center represented the class of programs that manipulate data bases. A third program combined these two types of applications. This program interrogated a data base for information concerning the traffic pattern at an airport and simulated future needs using a queuing algorithm.

These three programs were based on algorithms contained in Barrodale, Roberts, and Ehle (1971). The algorithms were modified

to incorporate only the constructs of sequence, structured iteration, and structured selection. They were then coded in Fortran and verified for correctness. Each of the resulting programs contained approximately 50 lines of executable code.

A section of about 15 lines was deleted from each program. This section, to be completed by the participants, was located somewhere near the middle of the program. The statements which the participants constructed consisted of assignment, selection, and iteration statements. All dimension, format, and input-output statements as well as all variable declarations were included in the participant's listing. The three programs are presented in Appendix A along with the constrained language - sequential specifications for each program. The programs are shown as they were presented to the participants (i.e., with the to-be-completed section deleted). For the reader's convenience, the corresponding portion of the specifications has been enclosed in brackets.

Type of Symbology. The statements from each program were translated into detailed specifications. Three types of symbology were used: natural language, constrained language, and ideograms. A consistent set of rules was used to map assignment, selection, and iteration statements across the three types of symbology.

Spatial Arrangements. Three spatial arrangements were used to represent the program structure: sequential, branching, and hierarchical. These three arrangements differed in the representation of control flow and nesting levels. In the sequential arrangement, both the control flow and the levels of nesting were represented vertically. In the branching arrangement, the flow of control was represented vertically while nesting levels were represented horizontally. Finally, in the hierarchical arrangement, the flow of control was represented horizontally while nesting levels were represented vertically.

Each of the three types of symbology was presented in the three spatial arrangements, resulting in nine specification formats for each program. Examples of the nine forms for the rocket trajectory program may be found in the first technical report of this series (Sheppard, Kruesi, and Curtis, 1980).

Procedure

Prior to the experiment, the participants were given a 20-minute training session in which they were shown each spatial arrangement and each type of symbology. The experimenter described the control flow for each arrangement using a sorting program as an example; this program was not seen in the actual experiment. The procedure for using the text editor to construct the programs was also explained in detail during the training session.

Experimental sessions were conducted at CRT terminals on a dedicated PDP-11/45. All coding was done in Fortran-IV. The participants were first given a practice program from which several lines had been deleted. Identical listings of the code appeared on the CRT screen and on a paper printout. The participants were instructed to complete the code, using the text editor. When satisfied that the program would perform correctly, the participants exited from the editor and activated a command file to compile and run the program. If the compilation was unsuccessful, a compiler message appeared on the screen directly below the line or lines containing the error. If the program had compiled, the output from the program appeared on the screen with one of the following messages: "OUTPUT IS CORRECT" or "OUTPUT IS INCORRECT." In the latter case, the participant was asked to correct the errors and submit the run again.

Following the practice program the three experimental programs were presented. For each program, the participants received one version of the specifications; these were contained on a single piece of paper. In addition, the participants received identical listings of the partially completed code on the CRT screen and on a paper printout. They also received a data dictionary containing the variable names, a natural language description of the variables, and the data types.

An interactive data collection system prompted the participant throughout the experimental procedure. The system recorded each call for an editor command (i.e., ADD, DELETE, LIST, CHANGE or RENUMBER) and the resultant changes in the program. An interval timer, accurate to the nearest second, recorded the time for each of these actions. When a participant required more than one editing session to complete the program correctly, the experimental system recorded exits from the editor, any compilation errors, and the incorrect outputs generated. From these data, the time to code and debug the programs was calculated by summing the times from the individual editing sessions; time for compiling and running the programs was not included. In addition, the participants' errors were identified and categorized as described in the Results section.

The participants spent approximately 25 minutes on each experimental program. They were required to continue working on a program until it was completed successfully. They were allowed to take breaks between programs.

Following the experiment, the participants completed a questionnaire about their previous programming experience. The information requested included number of years of professional experience, number of programming languages known, and whether

they had previously worked with algorithms of the type used in the experiment. The participants were also asked about their preferences for type of symbology and spatial arrangement.

Design

The three types of symbology (natural language, constrained language, and ideograms) were factorially combined with the three spatial arrangements (sequential, branching, and hierarchical) to produce nine specification formats. These nine formats were constructed for each of the three programs, resulting in a total of 27 conditions.

Participants received a set of specifications for each program. Across the three programs, they saw each type of symbology and each spatial arrangement. The first participant, for example, saw the rocket trajectory program presented in natural language - sequential, the inventory control program in constrained language - hierarchical, and the airport traffic program in ideograms - branching. The participants were assigned to conditions according to the procedures outlined in Winer (1971). [See also Kirk (1968)]. Each of the 27 conditions was used once within a set of nine participants. This 3^3 randomized block design provides a powerful assessment of the main effects and interactions. A minimum of 36 participants is required to assess all interactions and main effects. Across the 36 participants, each program, symbology, and arrangement was presented first, second, and third an equal number of times.

RESULTS

Time to Code and Debug

The participants required an average of 25 minutes to code and debug a program. This represents the amount of time spent using the text editor (i.e., the total time spent at the terminal less the time for compiling, linking and running).

There were large differences in the times required for the three programs (Table 1). The inventory program required the least time to complete (18.7 minutes); the airport program required the longest time (29.7 minutes).

Table 1 A Comparison of the Dependent Variables for the Three Programs

	PROGRAM			ALL PROGRAMS
	INVENTORY	ROCKET	AIRPORT	
MEAN TIME TO CODE AND DEBUG (MINUTES)	18.7	25.7	29.7	24.7
MEAN NUMBER OF SUBMISSIONS	1.8	2.8	3.5	2.7
MEAN NUMBER OF CONTROL-FLOW ERRORS	0.3	0.6	2.5	1.1
MEAN NUMBER OF ASSIGNMENT AND VARIABLE ERRORS	0.2	0.4	0.6	0.4
MEAN NUMBER OF EDITOR TRANSACTIONS	26.9	38.9	41.5	35.8

The difference among the programs was verified by an analysis of variance ($p<.001$). (See Table 2.) A stepwise multiple regression equation was used to partition the sums of squares for the ANOVA. Effects due to replications and participants within replications were removed first. Following that, effects due to the various

interactions were removed. Finally, main effects were accounted for. A logarithmic transformation was carried out on the times to attenuate the influence of extreme scores and to produce a more normal distribution (Kirk, 1968).

Table 2 Summary of ANOVA
Time to Code and Debug

SOURCE	*df*	*SS*	*MS*	*F*	ΔR^2	*p*
TOTAL	107	4.874				
BETWEEN PARTICIPANTS AND REPLICATIONS						
REPLICATIONS	3	.628			.13	
PARTICIPANTS WITHIN REPLICATIONS	32	1.076			.22	
WITHIN PARTICIPANTS AND REPLICATIONS						
PROGRAM (P)	2	.892	.446	16.76	.18	.001
SYMBOLOGY (S)	2	.445	.223	8.37	.09	.001
ARRANGEMENT (A)	2	.133	.067	2.50	.03	
P × S	4	.053	.013	.50	.01	
P × A	4	.177	.044	1.66	.04	
S × A	4	.184	.046	1.73	.04	
P × S × A	8	.063	.008	.30	.01	
RESIDUAL	46	1.224	.023			

Table 3 presents the code and debug times for each combination of symbology and spatial arrangement. The natural language versions required 29.7 minutes to complete, the ideograms required 23.9 minutes and the constrained language required 20.5 minutes. The effect of the type of symbology was highly significant ($p<.001$).

Table 3 Mean Time to Code and Debug (Minutes)

SPATIAL ARRANGEMENT	TYPE OF SYMBOLOGY			TOTAL
	NATURAL LANGUAGE	CONSTRAINED LANGUAGE	IDEOGRAMS	
SEQUENTIAL	31.8	16.5	26.6	25.0
BRANCHING	26.0	16.8	24.7	22.5
HIERARCHICAL	31.4	28.1	20.3	26.6
TOTAL	29.7	20.5	23.9	24.7

Note: Individual cell means represent 12 participants.

Differences due to the spatial arrangement were considerably smaller. Overall, the effect of spatial arrangement was not significant. There were no significant two or three way interactions. It is interesting to note that the constrained language presented in the sequential and branching arrangements led to the fastest times. A pairwise comparison revealed that differences among individual cell means greater than 6.9 minutes are significant at $p<.05$ and differences greater than 9.2 minutes are significant at $p<.01$.

Number of Submissions

The three programs were completed successfully by all participants. An average of 2.7 submissions were required to run the programs correctly. As with the code and debug times, there were substantial differences in the number of submissions across the three programs. As shown in Table 1, the inventory program was the least difficult, requiring 1.8 submissions and the airport program was the most difficult, requiring 3.5 submissions.

Table 4 presents the number of submissions broken down by the type of symbology and the spatial arrangement. The pattern shown parallels that for the code and debug times. The natural language

versions required 3.0 submissions, the ideograms required 2.7 and the constrained language required 2.2. Overall, the differences for the spatial arrangement are not pronounced. It is interesting to note that the constrained language presented in the sequential and branching arrangements required fewer submissions than the other versions.

Table 4 Mean Number of Submissions Required to Complete Task

SPATIAL ARRANGEMENT	TYPE OF SYMBOLOGY			TOTAL
	NATURAL LANGUAGE	CONSTRAINED LANGUAGE	IDEOGRAMS	
SEQUENTIAL	3.5	1.7	3.1	2.8
BRANCHING	2.5	1.8	2.9	2.4
HIERARCHICAL	3.1	3.2	2.2	2.8
TOTAL	3.0	2.2	2.7	2.7

Note: Individual cell means represent 12 participants.

Errors

The nature of the errors made by the participants provides valuable information about the difficulties they encountered in coding each program. No error analysis was attempted on data obtained prior to the first submission of a program. For programs that did not compile and run successfully on the first submission, the participants' editing activities for subsequent submissions were analyzed in detail to determine the nature of the errors.

The errors were assigned to two general categories: syntactic and semantic. The syntactic category includes a variety of errors that produced compiler messages. These errors were relatively easy to detect and correct. Unlike the semantic errors, the

syntactic errors could be corrected without reference to the specifications. Thus, they are of less interest than the semantic errors. The latter were divided into two subcategories: (1) control-flow errors and (2) assignment and variable errors. Appendix B contains a detailed breakdown of these subcategories.

Table 1 shows that there were large differences in the number of errors across the three programs. These differences follow the pattern previously shown for the times and the number of submissions. The inventory program was constructed with the fewest errors and the airport program showed the greatest number of errors. It is interesting to note that these differences reside almost entirely in the number of control-flow errors. The airport program resulted in considerably more of these errors (2.5) than either the inventory (0.3) or the rocket (0.6) program. Thus, the greater difficulty shown by the participants in coding the airport program resulted from problems with the control flow and not with assignment statements or variables.

A similar result occurred when the errors were examined as a function of the symbology and spatial arrangement. Differences among the nine formats were more pronounced for the control-flow errors than for the assignment and variable errors. As shown in Table 5, the branching arrangement resulted in far fewer control-flow errors than the sequential or hierarchical arrangements. It is interesting to note, however, that the fewest number of errors occurred with the constrained language - sequential format. As shown in Table 6, the assignment and variable errors were more evenly distributed across the nine formats. The major differences occurred within the sequential formats, with the natural language resulting in considerably more errors than the constrained language and ideograms.

Table 5 Mean Number of Control Flow Errors

SPATIAL ARRANGEMENT	TYPE OF SYMBOLOGY			TOTAL
	NATURAL LANGUAGE	CONSTRAINED LANGUAGE	IDEOGRAMS	
SEQUENTIAL	2.3	0.2	2.0	1.5
BRANCHING	0.3	0.4	0.3	0.3
HIERARCHICAL	2.5	1.2	0.9	1.5
TOTAL	1.7	0.6	1.1	1.1

Note: Individual cell means represent 12 participants.

Table 6 Mean Number of Assignment and Variable Errors

SPATIAL ARRANGEMENT	TYPE OF SYMBOLOGY			TOTAL
	NATURAL LANGUAGE	CONSTRAINED LANGUAGE	IDEOGRAMS	
SEQUENTIAL	0.9	0.1	0.2	0.4
BRANCHING	0.7	0.2	0.6	0.5
HIERARCHICAL	0.4	0.3	0.2	0.3
TOTAL	0.7	0.2	0.3	0.4

Note: Individual cell means represent 12 participants.

Editor Transactions

Another measure of the difficulty of using the specification formats is the number of commands entered at the terminal during the coding and debugging processes. The number of lines of code that were inserted into the program was added to the number of calls to the editor (e.g., ADD, DELETE, LIST, CHANGE or RENUMBER) to compute the total number of editor transactions that occurred. Table 1 compares the editor transactions for the three programs. The inventory program had the smallest number of editor transactions (26.9) and the airport program the largest number (41.5). An ANOVA (Table 7) confirmed the differences among programs ($p < .001$).

Table 7 Summary of ANOVA
Number of Editor Transactions
Occurring During Coding and Debugging

SOURCE	df	SS	MS	F	ΔR^2	p
TOTAL	107	33283.50				
BETWEEN PARTICIPANTS AND REPLICATIONS						
REPLICATIONS	3	305.63			.01	
PARTICIPANTS WITHIN REPLICATIONS	32	13789.22			.41	
WITHIN PARTICIPANTS AND REPLICATIONS						
PROGRAM (P)	2	4377.90	2188.95	11.62	.13	.001
SYMBOLOGY (S)	2	898.08	449.04	2.38	.03	
ARRANGEMENT (A)	2	76.35	38.18	0.20	.00	
P × S	4	1467.26	366.82	1.95	.05	
P × A	4	1371.04	342.76	1.82	.04	
S × A	4	1016.63	254.16	1.35	.03	
P × S × A	8	1317.68	164.71	0.87	.04	
RESIDUAL	46	8663.73	188.34			

The main effects for symbology and spatial arrangement were not statistically different (Table 7), and there were no significant two or three way interactions. However, a breakdown of the editor transactions by symbology and spatial arrangement (Table 8) shows that the sequential and branching versions of the constrained language required fewer editor transactions than the other versions. This pattern was found previously for the time to code and debug, for the number of submissions, and for the number of errors. A pairwise comparison test revealed that differences among individual cell means greater than 11 transactions were significant at $p<.05$, and differences greater than 15 transactions were significant at $p<.01$.

Table 8 Mean Number of Editor Transactions Occurring During Coding and Debugging

SPATIAL ARRANGEMENT	TYPE OF SYMBOLOGY			TOTAL
	NATURAL LANGUAGE	CONSTRAINED LANGUAGE	IDEOGRAMS	
SEQUENTIAL	43	26	41	37
BRANCHING	34	29	41	35
HIERARCHICAL	34	40	35	36
TOTAL	37	32	39	36

Note: Individual cell means represent 12 participants.

Preferences for Type of Symbology and Spatial Arrangement

Across the three programs, each participant received specifications in each type of symbology and in each spatial arrangement. On the questionnaire, they were asked to state which symbology and which arrangement they preferred. Table 9 shows these preferences.

Table 9 Percent of Preferences for Symbology and Spatial Arrangement

FACTOR	%
TYPE OF SYMBOLOGY :	
CONSTRAINED LANGUAGE	59
IDEOGRAMS	35
NATURAL LANGUAGE	6
SPATIAL ARRANGEMENT:	
SEQUENTIAL	23
BRANCHING	62
HIERARCHICAL	15

In terms of the type of symbology, the majority of participants chose the constrained language, ideograms were intermediate, while natural language was the least preferred. In terms of the spatial arrangement, branching was the most preferred, sequential was intermediate and hierarchical was the least preferred.

Experiential Factors

The participants were asked the number of years they had programmed professionally and the number of programming languages they knew. No correlation was found between time to code and debug and these experiential factors.

DISCUSSION

The previous experiment in this series involved a comprehension task and employed the specification formats that were used in the current experiment. The present results closely parallel those from the first experiment. The discussion below describes each conclusion drawn from this experiment and compares it to the corresponding conclusion from the comprehension experiment.

Substantial differences were found among the three programs. The airport program was the most difficult as measured by all dependent variables (the time required to code and debug, and the number of submissions, errors, and editor transactions). The inventory program was the least difficult. A similar pattern of program difficulty was found in the comprehension experiment, although the differences in that experiment were much less pronounced. A question of interest concerns why the airport program was so much more difficult than either the inventory or rocket programs. The sections that were completed for the three programs had been chosen because of their comparable size and control flow complexity. Each consisted of the major portion of a DO WHILE loop. A detailed examination of the error data revealed two explanations for the greater difficulty of the airport program. Both involved the control flow. In the inventory and rocket programs, the statement which directed the return to the top of the loop was provided to the participants (See Appendix A). This statement was not provided for the airport program and was the greatest source of errors: one-third of the participants did not provide the return statement before their first submission. Other frequent errors for the airport program were contained in the control paths for the three nested IF statements. This control structure was more deeply nested than those of the other two programs.

Differences among the three programs may also be accounted for, in part, by the previous programming experience of the participants. When asked whether they had worked with programs of a given type, 44% responded that they had worked with an inventory control program while only 14% and 17% respectively had worked with airport and rocket programs.

Substantial differences were also associated with the type of symbology. The natural language versions resulted in longer times, more submissions, more errors, and more editor transactions than the constrained language and ideogram versions. The natural language - sequential version was a particularly difficult format for the participants to use. The same pattern was found in the comprehension experiment.

Had the natural language been written casually, one could hypothesize that it was incomplete and misleading. However, the natural language in this experiment was developed very precisely. Assignment, selection and iteration statements were translated from the original code into the three types of symbology according to a rigid set of rules to insure that the natural language specifications were as complete and precise as the constrained language and ideograms. It is reasonable to conclude, therefore, that the differences were due to real differences among the types of symbology rather than to an experimental artifact. In real systems development, this effect may be even more pronounced since it is unlikely that the specifications are as carefully developed.

The longer response times for natural language might be expected because the coder's task involved an extra procedure not necessary for the constrained language and ideogram versions. To translate from the natural language to Fortran, the coder had to inspect the data dictionary, matching the natural language

description on the specification sheet to the natural language description on the data dictionary thus retrieving the appropriate variable name. However, the increases in the natural language times were greater than one would expect from performing the extra table look-up procedure. The natural language - sequential version took almost twice as long as the constrained language - sequential version (31.8 vs. 16.5 minutes). More importantly the natural language - sequential was associated with 3.2 semantic errors per program as opposed to 0.3 error for the constrained language - sequential. A high proportion of the natural language - sequential errors were related to the control flow (2.3 errors per program). Only 0.9 error per program was associated with assignment statements and variable names, indicating that retrieval of the variable names from the data dictionary was not the principal cause of the errors. The distinction between these versions is relevant because many programming installations use natural language - sequential when producing their specifications.

The effect of spatial arrangement was not as great as that of symbology. This result was found in the comprehension experiment. Although not statistically significant, the branching arrangement appeared to be mildly superior to the sequential and hierarchical arrangements, particularly in reducing the number of errors related to the control flow. This is not a suprising result. Programmers have normally used flowcharts, the ideogram - branching format, to solve complicated control-flow problems. The interesting result is that the branching arrangement can be combined with the constrained language to produce a format that can compete favorably with flowcharts.

The two formats which led to consistently superior coding performance were the constrained language - sequential (normal PDL) and the constrained language - branching. Both formats also led to superior performance in the comprehension experiment. The

ideogram - branching format (normal flowchart) appeared as good as the other two formats in the comprehension experiment but not in the current experiment.

The preferences of the participants closely paralleled those found in the comprehension experiment. In both experiments, constrained language was the preferred symbology and branching was the preferred spatial arrangement.

The number of years of programming experience did not relate to performance. This result was also found in the comprehension experiment. One discrepancy between the two experiments lies in the predictive value of the diversity of the participants' experience as measured by the number of programming languages. Diversity was significantly related to performance in the comprehension experiment but not in the current experiment. This reason for this discrepancy is not clear.

This experiment provides additional evidence that specification format can have a significant effect on the performance of programmers on software-related tasks. A coding task was carried out more quickly and with fewer errors from specifications presented in a succinct symbology. An examination of the individual cell means revealed that the constrained language presented in a sequential or in a branching arrangement led to the highest level of performance. In terms of the choices of specification formats currently in use, software developers would be well advised to convert software specifications to a PDL before coding begins.

ACKNOWLEDGEMENTS

The authors would like to thank Dave Morris and Pete McEvoy for designing the automatic data collection system, Carol Kiefer, Joan Shields, Leo Pompliano and Gene Poggenburg for providing participants, John O'Hare, Bill Curtis and John Bailey for their advice, and Tom McDonald for preparing materials and statistical analyses.

REFERENCES

Barrodale, I., Roberts, F.D.K., & Ehle, B.L. Elementary computer applications in science, engineering, and business. New York: Wiley, 1971.

Blaiwes, A.S. Formats for presenting procedural instructions. Journal of Applied Psychology, 1974, 59, 683-686.

Bohl, M. Flowcharting techniques. Palo Alto, CA: Science Research Associates, 1971.

Boies, S.J. & Gould, J.D. Syntactic errors in computer programming. Human Factors, 1974, 16, 253-257.

Brooks, F.P. The mythical man-month. Reading, MA: Addison-Wesley, 1975.

Caine, S.H. & Gordon, E.K. PDL - A tool for software design. In Proceedings of the 1975 National Computer Conference. Montvale, NJ: American Federation of Information Processing Societies, 1975.

Dunsmore, H.E. & Gannon, J.D. The effect of individual programmer tendencies on the program construction process. (Unpublished manuscript). College Park, MD: Department of Computer Science, University of Maryland, 1978.

Jones, C. A Survey of programming design and specification techniques. In Proceedings of the IEEE Conference on Specifications of Reliable Software. New York: Institute of Electrical and Electronics Engineers, 1979.

Kammann, R. The comprehensibility of printed instructions and the flowchart alternative. Human Factors, 1975, 17, 183-191.

Katzen, H. Systems design and documentation: An introduction to the HIPO method. New York: Van Nostrand Reinhold, 1976.

Kirk, R.E. Experimental design procedures for the behavioral sciences. Belmont, CA: Brooks-Cole, 1968.

Lucas, H.C. & Kaplan, R.B. A structured programming experiment. The Computer Journal, 1974, 19, 136-138.

Ramsey, H.R., Atwood, M.E., & Van Doren, J.R. A comparative study of flowcharts and program design languages for the detailed procedural specification of computer programs. (Tech. Rep. #SAI-78-078-DEN). Denver: Science Applications, Inc. 1978.

Sheppard, S.B., Curtis, B., Milliman, P., & Love, T. Modern coding practices and programmer performance. Computer, 1979, 12, (12), 41-49.

Sheppard, S.B., Kruesi, E., & Curtis, B. The effects of symbology and spatial arrangement on the comprehension of software specifications (Tech. Rep. TR-80-388200-2). Arlington, VA: General Electric, Information Systems Programs, 1980.

Sheppard, S.B., Milliman, P., & Curtis, B. Experimental evaluation of on-line program construction (Tech. Rep. TR-79-388100-6). Arlington, VA: General Electric, Information Systems Programs, 1979.

Shneiderman, B., Mayer, B.R., McKay, D., & Heller, P. Experimental investigations on the utility of detailed flowcharts in programming. Communications of the ACM, 1977, 20, 373-381.

Sime, M.E., Green, T.R.G., & Guest, D.J. Scope marking in computer conditionals - A psychological evaluation. International Journal of Man-Machine Studies, 1977, 9, 107-118.

Winer, B.J. Statistical principles in experimental design. New York: McGraw-Hill, 1971.

Wright, P. & Reid, F. Written information: Some alternatives to prose for expressing the outcomes of complex contingencies. Journal of Applied Psychology, 1973, 57, 160-166.

Youngs, E.A. Human errors in programming. International Journal of Man-Machine Studies, 1974, 6, 361-376.

APPENDIX A

Program Listings as Given to Participants and Constrained Language - Sequential Specifications

INVENTORY PROBLEM

```
      INTEGER DELIV, FLAG, ITEM, ONHAND, ORDER, RELEV, REORD,
     1 STORE, UNFILL
      REAL GTOTAL, PRICE, TOTAL
      OPEN (UNIT=1, NAME='ORDERS.DAT', TYPE='OLD')
      OPEN (UNIT=2, NAME='PURCHAS.DAT', TYPE='OLD',
     1 ACCESS='SEQUENTIAL')
   10 READ (1, 100, END=80) STORE
      GTOTAL = 0
      WRITE (5, 110) STORE
   20 READ (1, 120) ITEM, ORDER
      IF (ITEM .EQ. 0) GO TO 70
      CALL FETCH2 (ITEM, PRICE, ONHAND, RELEV, REORD, FLAG)
C               YOUR CODE BEGINS HERE
C
C
C
      IF (FLAG .NE. 1) GO TO 60
      WRITE (2, 130) ITEM, REORD
      FLAG = 2
   60 WRITE (5, 140) ITEM, PRICE, ORDER, DELIV, UNFILL, TOTAL
      CALL UPDATE (ITEM, ONHAND, FLAG)
      GO TO 20
   70 WRITE (5, 150) GTOTAL
      GO TO 10
   80 CLOSE (UNIT=1)
      CLOSE (UNIT=2)
      STOP
  100 FORMAT (I2)
  110 FORMAT (//, 5X, 'INVOICE FOR STORE NUMBER:', I3)
  120 FORMAT (I3, I5)
  130 FORMAT (2I7)
  140 FORMAT (5X, 'ITEM NUMBER:', I11 / 5X,
    1 'PRICE PER ITEM:  $', F5.2 / 5X, 'NUMBER ORDERED:', I8 /
    1 5X, 'NUMBER DELIVERED:', I6 / 5X, 'UNABLE TO DELIVER:', I5 /
    1 5X, 'TOTAL PRICE:  $', F8.2)
  150 FORMAT (/, 5X, 'TOTAL PRICE FOR ALL ITEMS:  $', F10.2)
      END
```

```
      INTEGER MAXT,TIME,FLAG
      REAL VACCEL,VVELOC,VDIST,HACCEL,HVELOC,HDIST,
     1 ANGLE,TILT,GRAV,MASS,FUEL,FORCE
      VACCEL = 0.
      VVELOC = 0.
      VDIST = 0.
      HACCEL = 0.
      HVELOC = 0.
      HDIST = 0.
      ANGLE = 0.
      TILT = 0.3491
      GRAV = 32.
      MASS = 10000.
      FUEL = 50.
      FORCE = 400000.
      MAXT = 200
      FLAG = 0
      TIME = 1
   10 IF (FLAG .NE. 0) GO TO 60
C              YOUR CODE BEGINS HERE
C
C
C
      GO TO 10
   60 TIME = TIME + 1
      IF (VDIST .GT. 0) GO TO 80
   70 WRITE(5,3000) TIME, HDIST
      GO TO 90
   80 WRITE(5,4000) TIME, MASS, VACCEL, VVELOC, VDIST,
     1 HACCEL,HVELOC,HDIST
   90 CONTINUE
      STOP
 3000 FORMAT(5X,'ROCKET HIT GROUND AT TIME =',I5,' SECONDS'/
     1 5X,'HORIZONTAL DIST = ',F11.2)
 4000 FORMAT(5X,'ROCKET STILL ALOFT AT TIME = ',I5,' SECONDS'/
     1 5X,'MASS = ',F22.2/
     2 5X,'VERTICAL ACCEL = ',F12.2/
     3 5X,'VERTICAL VELOC = ',F12.2/
     4 5X,'VERTICAL DIST = ',F13.2/
     5 5X,'HORIZONTAL ACCEL = ',F10.2/
     6 5X,'HORIZONTAL VELOC = ',F10.2/
     7 5X,'HORIZONTAL DIST = ',F11.2)
      END
```

AIRPORT PROBLEM

```
          INTEGER ARRQUE, BEGINT, CLEAR, DEPQUE, ENDT, MAXWT,
     1    NUMARR, NUMDEP, TIME, TOLWT, R
          REAL ARPROB, DPPROB, RAND1, RAND2
          R = 0
          CALL FETCH1 (BEGINT, ARPROB, DPPROB, ARRQUE, DEPQUE,
     1    CLEAR, TOLWT)
          NUMARR = 0
          NUMDEP = 0
          TIME = BEGINT
          ENDT = BEGINT + 20
   10     IF (TIME .GT. ENDT) GO TO 60
          RAND1 = RND(R)
          IF (RAND1 .GT. ARPROB) GO TO 20
          ARRQUE = ARRQUE + 1
   20     RAND2 = RND(R)
          IF (RAND2 .GT. DPPROB) GO TO 30
          DEPQUE = DEPQUE + 1
   30     CONTINUE
C
C                    YOUR CODE BEGINS HERE
C
C

          WRITE (5, 100) ENDT, ARRQUE, NUMARR, DEPQUE, NUMDEP, MAXWT
          IF (MAXWT .GT. TOLWT) GO TO 70
          WRITE (5, 120)
          GO TO 80
   70     WRITE (5, 110)
   80     CONTINUE
          STOP
  100     FORMAT (5X, 'ENDING TIME FOR SIMULATION:', I5,/,
     1    12X, 'ARRIVAL QUEUE:', I5/11X, 'NUMBER ARRIVED:', I5/
     1    10X, 'DEPARTURE QUEUE:', I5/10X, 'NUMBER DEPARTED:', I5/
     1    13X, 'MAXIMUM WAIT:', I5, ' MINUTES')
  110     FORMAT (5X, 'OPEN ANOTHER RUNWAY')
  120     FORMAT (5X, 'ANOTHER RUNWAY NOT NEEDED')
          END
```

```
PROGRAM INVENTORY

READ FROM 'ORDERS':  STORE

DO WHILE NOT END OF 'ORDERS'

     SET GTOTAL = 0

     PRINT 'INVOICE FOR STORE NUMBER', STORE

     READ FROM 'ORDERS':  ITEM, ORDER

     DO WHILE ITEM ≠ 0

          FETCH FROM DATA BASE FOR ITEM:  PRICE, ONHAND, RELEV, REORD, FLAG

          IF ONHAND > ORDER

               THEN

               SET DELIV  = ORDER
               SET ONHAND = ONHAND-ORDER
               SET UNFILL = 0

               ELSE

               SET DELIV  = ONHAND
               SET ONHAND = 0
               SET UNFILL = ORDER-DELIV

          ENDIF

          IF ONHAND ≤ RELEV

               THEN

               IF FLAG = 0

                    THEN

                    SET FLAG = 1

               ENDIF

          ENDIF

          SET TOTAL  = DELIV* PRICE
          SET GTOTAL = GTOTAL + TOTAL

          IF FLAG = 1
```

```
            ENDIF

            SET TOTAL  = DELIV* PRICE
            SET GTOTAL = GTOTAL + TOTAL

            IF FLAG = 1

                THEN

                WRITE TO 'PURCHAS':  ITEM, REORD

                SET FLAG = 2

            ENDIF

            PRINT ITEM, PRICE, ORDER, DELIV, UNFILL, TOTAL

            UPDATE DATA BASE FOR ITEM:  ONHAND, FLAG

            READ FROM 'ORDERS':  ITEM, ORDER

        ENDDO

        PRINT GTOTAL

        READ FROM 'ORDERS':  STORE

    ENDDO

    END OF INVENTORY
```

```
PROGRAM ROCKET

SET VACCEL = 0
SET VVELOC = 0
SET VDIST  = 0
SET HACCEL = 0
SET HVELOC = 0
SET HDIST  = 0
SET ANGLE  = 0
SET TILT   = 0.3491
SET GRAV   = 32
SET MASS   = 10000
SET FUEL   = 50
SET FORCE  = 400000
SET MAXT   = 200

SET FLAG   = 0
SET TIME   = 1

DO WHILE FLAG = 0

     IF TIME ≤ 100

          THEN

          SET MASS = MASS-FUEL

          IF TIME = 11

               THEN

               SET ANGLE = TILT

          ENDIF

          ELSE

          IF TIME = 101

               THEN

               SET FORCE = 0

          ENDIF

     ENDIF

     SET VACCEL = ((FORCE * COS(ANGLE))/MASS) - GRAV
     SET VVELOC =  VVELOC + VACCEL
     SET VDIST  =  VDIST + VVELOC
     SET HACCEL = (FORCE * SIN(ANGLE))/MASS
     SET HVELOC =  HVELOC + HACCEL
     SET HDIST  =  HDIST + HVELOC

     SET TIME = TIME + 1

     IF VDIST ≤ 0

          THEN
```

```
    SET HDIST = HDIST + HVELOC
    SET TIME = TIME + 1
    IF VDIST ≤ 0
        THEN
        SET FLAG = 1
    ENDIF
    IF TIME > MAXT
        THEN
        SET FLAG = 2
    ENDIF
ENDDO
SET TIME = TIME - 1
IF VDIST > 0
    THEN
    PRINT "ROCKET STILL ALOFT", TIME, MASS, VACCEL,
    VVELOC, VDIST, HACCEL, HVELOC, HDIST
    ELSE
    PRINT "ROCKET HIT GROUND", TIME, HDIST
ENDIF
END OF ROCKET
```

```
PROGRAM AIRPORT

FETCH FROM DATA BASE:  BEGINT, ARPROB, DPPROB, ARRQUE, DEPQUE, CLEAR, TOLWT

SET NUMARR = 0
SET NUMDEP = 0
SET TIME   = BEGINT
SET ENDT   = BEGINT + 20

DO WHILE TIME ≤ ENDT

     SET RAND1 = RND(R)

     IF RAND1 ≤ ARPROB

          THEN

          SET ARRQUE = ARRQUE + 1

     ENDIF

     SET RAND2 = RND(R)

     IF RAND2 ≤ DPPROB

          THEN

          SET DEPQUE = DEPQUE + 1

     ENDIF

     IF CLEAR ≤ TIME

          THEN

          IF ARRQUE > 0

               THEN

               SET ARRQUE = ARRQUE - 1
               SET NUMARR = NUMARR + 1
               SET CLEAR  = TIME + 3

               ELSE

               IF DEPQUE > 0

                    THEN

                    SET DEPQUE = DEPQUE - 1
                    SET NUMDEP = NUMDEP + 1
                    SET CLEAR  = TIME + 2
```

```
                        THEN

                        SET DEPQUE = DEPQUE - 1
                        SET NUMDEP = NUMDEP + 1
                        SET CLEAR  = TIME + 2

                    ENDIF

                ENDIF

            ENDIF

            SET TIME = TIME + 1

        ENDDO

        SET MAXWT = (CLEAR - ENDT) + (ARRQUE * 3) + (DEPQUE * 2)

        PRINT ENDT, ARRQUE, NUMARR, DEPQUE, NUMDEP, MAXWT

        IF MAXWT > TOLWT

            THEN

            PRINT "OPEN ANOTHER RUNWAY"

            ELSE

            PRINT "ANOTHER RUNWAY NOT NEEDED"

        ENDIF

        END OF AIRPORT
```

APPENDIX B

Errors by Type of Symbology
and Spatial Arrangement

NUMBER OF ERRORS BY TYPE OF SYMBOLOGY AND SPATIAL ARRANGEMENT

	NATURAL LANGUAGE			CONSTRAINED LANGUAGE			IDEOGRAMS			
CONTROL FLOW ERRORS	SEQ	BR	HIER	SEQ	BR	HIER	SEQ	BR	HIER	
INCORRECT LOGICAL OPERATOR	4	0	8	0	4	2	2	1	5	26
INCORRECT "GO TO #"	6	1	5	1	0	1	10	0	0	24
MISSING "GO TO" STATEMENT	8	1	7	1	1	5	7	1	2	33
INCORRECT STATEMENT LABEL	3	1	2	0	0	1	1	1	0	9
MISSING STATEMENT LABEL	4	1	5	1	0	5	3	1	3	23
EXTRA "IF" OR "GO TO" STATEMENT	3	0	3	0	0	1	1	0	1	9
TOTAL CONTROL FLOW ERRORS	28	4	30	3	5	15	24	4	11	124
ASSIGNMENT OR VARIABLE ERRORS										
STATEMENT MISSING (OTHER THAN "GO TO")	6	1	0	1	1	1	2	2	0	14
INCORRECT ARITHMETIC OPERATOR	3	0	0	0	0	0	0	2	2	7
INCORRECT CONSTANT	0	0	1	0	1	2	0	2	0	6
INCORRECT VARIABLE NAME	2	6	2	0	1	0	0	1	0	12
EXTRA VARIABLES OR INCORRECT ORDER	0	1	2	0	0	1	0	0	0	4
TOTAL ASSIGNMENT OR VARIABLE ERRORS	11	8	5	1	3	4	2	7	2	43
SYNTAX ERRORS	13	14	7	2	9	12	7	3	8	75
TOTAL ERRORS	52	26	42	6	17	31	33	14	21	242

Technical Reports Distribution List

OFFICE OF NAVAL RESEARCH

Code 455

TECHNICAL REPORTS DISTRIBUTION LIST

OSD

CDR Paul R. Chatelier
Office of the Deputy Under Secretary
of Defense
OUSDRE (E&LS)
Pentagon, Room 3D129
Washington, D.C. 20301

Department of the Navy

Director
Engineering Psychology Programs
Code 455
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217 (5 cys)

Director
Aviation & Aerospace Technology
Code 210
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Director
Undersea Technology
Code 220
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Director
Electronics & Electromagnetics
Technology
Code 250
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Director
Communication & Computer Technology
Code 240
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Department of the Navy

Director
Tactical Development & Evaluation
Support
Code 230
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Director
Operations Research Programs
Code 434
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Director
Statistics and Probability Program
Code 436
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Director
Information Systems Program
Code 437
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Director
Physiology Program
Code 441
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Special Assistant for Marine
Corps Matters
Code 100M
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Department of the Navy

Commanding Officer
ONR Eastern/Central Regional Office
ATTN: Dr. J. Lester
Building 114, Section D
666 Summer Street
Boston, MA 02210

Commanding Officer
ONR Branch Office
ATTN: Dr. C. Davis
536 South Clark Street
Chicago, IL 60605

Commanding Officer
ONR Western Regional Office
ATTN: Mr. R. Lawson
1030 East Green Street
Pasadena, CA 91106

Commanding Officer
ONR Western Regional Office
ATTN: Dr. E. Gloye
1030 East Green Street
Pasadena, CA 91106

Office of Naval Research
Scientific Liaison Group
American Embassy, Room A-407
APO San Francisco, CA 96503

Director
Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375 (6 cys)

Dr. Bruce Wald
Communications Sciences Division
Code 7500
Naval Research Laboratory
Washington, D.C. 20375

K. L. Heninger
Code 7503
Naval Research Laboratory
Washington, D.C. 20375

Department of the Navy

Dr. Robert G. Smith
Office of the Chief of Naval Operations, OP987H
Personnel Logistics Plans
Washington, D.C. 20350

Naval Training Equipment Center
ATTN: Technical Library
Orlando, FL 32813

Human Factors Department
Code N215
Naval Training Equipment Center
Orlando, FL 32813

Dr. Alfred F. Smode
Training Analysis and Evaluation Group
Naval Training Equipment Center
Code N-OOT
Orlando, FL 32813

CDR R. Gibson
Bureau of Medicine & Surgery
Aerospace Psychology Branch
Code 513
Washington, D.C. 20372

CDR Robert Biersner
Naval Medical R&D Command
Code 44
Naval Medical Center
Bethesda, MD 20014

Dr. Arthur Bachrach
Behavioral Sciences Department
Naval Medical Research Institute
Bethesda, MD 20014

CDR Thomas Berghage
Naval Health Research Center
San Diego, CA 92152

Dr. George Moeller
Human Factors Engineering Branch
Submarine Medical Research Lab
Naval Submarine Base
Groton, CT 06340

Department of the Navy

Head
Aerospace Psychology Department
Code L5
Naval Aerospace Medical Research Lab
Pensacola, FL 32508

J. B. Blankenheim
Code 47013
Naval Electronic Systems Command
NC Bldg #1, RM 4E40
Washington, D.C. 20360

Dr. James McGrath, Code 302
Navy Personnel Research and
Development Center
San Diego, CA 92152

Navy Personnel Research and
Development Center
Planning & Appraisal
Code 04
San Diego, CA 92152

Navy Personnel Research and
Development Center
Management Systems, Code 303
San Diego, CA 92152

Navy Personnel Research and
Development Center
Performance Measurement &
Enhancement
Code 309
San Diego, CA 92152

CDR P. M. Curran
Code 604
Human Factors Engineering Division
Naval Air Development Center
Warminster, PA 18974

Dean of the Academic Departments
U.S. Naval Academy
Annapolis, MD 21402

Dr. Gary Poock
Operations Research Department
Naval Postgraduate School
Monterey, CA 93940

Department of the Navy

Dean of Research Administration
Naval Postgraduate School
Monterey, CA 93940

Mr. Warren Lewis
Human Engineering Branch
Code 8231
Naval Ocean Systems Center
San Diego, CA 92152

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Mr. Arnold Rubinstein
Naval Material Command
NAVMAT 0722 - Rm. 508
800 North Quincy Street
Arlington, VA 22217

Commander
Naval Air Systems Command
Human Factors Programs
NAVAIR 340F
Washington, D.C. 20361

Commander
Naval Air Systems Command
Crew Station Design,
NAVAIR 5313
Washington, D.C. 20361

Mr. Phillip Andrews
Naval Sea Systems Command
NAVSEA 0341
Washington, D.C. 20362

Commander
Naval Electronics Systems Command
Human Factors Engineering Branch
Code 4701
Washington, D.C. 20360

Department of the Navy

Human Factors Section
Systems Engineering Test
Directorate
U.S. Naval Air Test Center
Patuxent River, MD 20670

Mr. John Impagliazzo, Code 101
Newport Laboratory
Naval Underwater Systems Center
Newport, RI 02840

LCDR W. Moroney
Code 55MP
Naval Postgraduate School
Monterey, CA 93940

Mr. Merlin Malehorn
Office of the Chief of Naval
Operations (OP-115)
Washington, D.C. 20350

Department of the Army

Mr. J. Barber
HQS, Department of the Army
DAPE-MBR
Washington, D.C. 20310

Dr. Joseph Zeidner
Technical Director
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Director, Organizations and
Systems Research Laboratory
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Technical Director
U.S. Army Human Engineering Labs
Aberdeen Proving Ground, MD 21005

U.S. Army Medical R&D Command
ATTN: CPT Gerald P. Krueger
Ft. Detrick, MD 21701

Department of the Army

ARI Field Unit-USAREUR
ATTN: Library
C/O ODCSPER
HQ USAREUR & 7th Army
APO New York 09403

Department of the Air Force

U.S. Air Force Office of Scientific
Research
Life Sciences Directorate, NL
Bolling Air Force Base
Washington, D.C. 20332

Dr. Donald A. Topmiller
Chief, Systems Engineering Branch
Human Engineering Division
USAF AMRL/HES
Wright-Patterson AFB, OH 45433

Air University Library
Maxwell Air Force Base, AL 36112

Dr. Gordon Eckstrand
AFHRL/ASM
Wright-Patterson AFB, OH 45433

Dr. Earl Alluisi
Chief Scientist
AFHRL/CCN
Brooks AFB, TX 78235

Foreign Addressees

North East London Polytechnic
The Charles Myers Library
Livingstone Road
Stratford
London E15 2LJ
ENGLAND

Professor Dr. Carl Graf Hoyos
Institute for Psychology
Technical University
8000 Munich
Arcisstr 21
FEDERAL REPUBLIC OF GERMANY

Foreign Addressees

Dr. Kenneth Gardner
Applied Psychology Unit
Admiralty Marine Technology
Establishment
Teddington, Middlesex TW11 OLN
ENGLAND

Director, Human Factors Wing
Defence & Civil Institute of
Environmental Medicine
Post Office Box 2000
Downsview, Ontario M3M 3B9
CANADA

Dr. A. D. Baddeley
Director, Applied Psychology Unit
Medical Research Council
15 Chaucer Road
Cambridge, CB2 2EF
ENGLAND

Other Government Agencies

Defense Technical Information Center
Cameron Station, Bldg. 5
Alexandria, VA 22314 (12 cys)

Dr. Craig Fields
Director, Cybernetics Technology
Office
Defense Advanced Research Projects
Agency
1400 Wilson Blvd
Arlington, VA 22209

Dr. M. Montemerlo
Human Factors & Simulation
Technology, RTE-6
NASA HQS
Washington, D.C. 20546

Other Organizations

Dr. H. McI. Parsons
Human Resources Research Office
300 N. Washington Street
Alexandria, VA 22314

Other Organizations

Dr. Jesse Orlansky
Institute for Defense Analyses
400 Army-Navy Drive
Arlington, VA 22202

Dr. Arthur I. Siegel
Applied Psychological Services, Inc.
404 East Lancaster Street
Wayne, PA 19087

Dr. Robert T. Hennessy
NAS - National Research Council
JH #819
2101 Constitution Ave., N.W.
Washington, DC 20418

Dr. M. G. Samet
Perceptronics, Inc.
6271 Variel Avenue
Woodland Hills, CA 91364

Dr. Robert Williges
Human Factors Laboratory
Virginia Polytechnical Institute
and State University
130 Whittemore Hall
Blacksburg, VA 24061

Dr. Timothy Lindquist
Department of Computer Science
VPI & SU
Blacksburg, VA 24061

Dr. Alphonse Chapanis
Department of Psychology
The Johns Hopkins University
Charles and 34th Streets
Baltimore, MD 21218

Dr. Meredith P. Crawford
American Psychological Association
Office of Educational Affairs
1200 17th Street, NW.
Washington, D.C. 20036

Dr. James H. Howard, Jr.
Department of Psychology
Catholic University
Washington, D.C. 20064

Other Organizations

Journal Supplement Abstract Service
American Psychological Association
1200 17th Street, N.W.
Washington, D.C. 20036 (3 cys)

Mr. Edward M. Connelly
Performance Measurement
Associates, Inc.
131 Park Street, NW.
Vienna, VA 22180

Dr. Edward R. Jones
Chief, Human Factors Engineering
McDonnell-Douglas Astronautics
Company
St. Louis Division
Box 516
St. Louis, MO 63166

Dr. David J. Getty
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

Dr. Douglas Towne
University of Southern California
Behavioral Technology Laboratory
3716 S. Hope Street
Los Angeles, CA 90007

F

4